WHAT IS CLAIMED IS:

1. An execution unit for execution of multiple context threads comprises:
an arithmetic logic unit to process data for executing threads;
control logic to control the operation of the arithmetic logic unit;
a general purpose register set to store and obtain operands for the arithmetic logic unit, the register set constructed with a two-ported random access memory, with the register set divided into a plurality of banks and with two different words from each bank of the register set can be read and written in the same processor cycle.

2. The execution unit of claim 1 wherein the register set is logically partitioned into a plurality of relatively addressable windows.

3. The execution unit of claim 2 wherein the number of windows of the register set is according to the number of threads that can execute in the processor.

4. The execution unit of claim 1 where the relative addressing allows the currently executing thread to access to any of the registers relative to the starting point of a window of registers.

5. The execution unit of claim 1 wherein the register set is absolutely addressable where any one of the addressable registers may be accessed by the currently executing thread by providing the exact address of the register.

6. The execution unit of claim 1 wherein the control logic further comprises: context event switching logic fed by signals from a plurality of shared resources with the signals causing the context event logic to indicate that threads are either available or unavailable for execution.

7. The execution unit of claim 6 wherein the control logic addresses a set of memory locations for storing a list of available threads that correspond to threads that are ready to be executed and a set of memory locations for storing a list of unavailable threads that are not ready to be executed.

8. The execution unit of claim 7 wherein execution of a context swap instruction causes a currently running thread to be swapped out to the unavailable thread memory set and a thread from the available thread memory set to begin execution within a single execution cycle.

9. The execution unit of claim 8 wherein execution of the context swap instruction specifies one of the signal inputs and upon receipt of the specified signal input causes the swapped out thread to be stored in the available thread memory set.

10. The execution unit of claim 8 wherein execution of the context swap instruction specifies a sequence number change and upon receipt of the specified sequence number change causes the swapped out thread to be stored in the unavailable memory set.

11. The execution unit of claim 8 wherein execution of the context swap instruction specifies an inter-thread signal input and upon receipt of the specified inter-thread signal causes the swapped out thread to be stored in the available memory set.

12. The execution unit of claim 8 wherein execution of the context swap instruction specifies a voluntary swap operation and causes a context swap if there is a thread in the available memory set ready to be executed.

13. The execution unit of claim 8 wherein execution of the context swap instruction specifies a defer_one operation which causes execution of one more instruction and then causes the current context to be swapped out.

14. The execution unit of claim 8 wherein the context event switching logic further comprises:

a ctx_enable register and with execution of the context swap instruction specifying a kill operation causing the ctx_enable bit to be set to indicate that this thread is not available for execution until the bit is cleared by another instruction.

15. A method for executing multiple context threads comprises:

processing data for executing threads within an arithmetic logic unit;

operating control logic to control the arithmetic logic unit;

storing and obtaining operands for the arithmetic logic unit within a general purpose register with the register set constructed with a two-ported random access memory; and

arranging the register set into a plurality of banks where two separate words from separate banks of the register set can be read and written in the same processor cycle.

16. The method of claim 15 wherein the register is relatively addressable.

17. The method of claim 15 further comprising:

arranging the register set into a number of windows according to the number of threads that can execute in the processor.

18. The method of claim 17 wherein storing and obtaining further comprises:

addressing the registers by the currently executing thread by providing a relative register address that is relative to the starting point of a window of registers.

19. A processor unit comprises:

an execution unit for execution of multiple context threads comprising:

an arithmetic logic unit to process data for executing threads;

control logic to control the operation of the arithmetic logic unit;

a general purpose register set to store and obtain operands for the arithmetic logic unit, the register set constructed with a two-ported random access memory.

20. The processor of claim 19 wherein the register set is logically partitioned into a plurality of relatively addressable windows where the number of windows of the register set is according to the number of threads that can execute in the processor.

21. The processor of claim 20 where the relative addressing allows the currently executing thread to access to any of the registers relative to the starting point of a window of registers.

22 The processor of claim 20 wherein the register set is absolutely addressable where any one of the addressable registers may be accessed by the currently executing thread by providing the exact address of the register.

23. The processor of claim 20 further comprises:

a set of memory locations for storing a list of available threads that correspond to threads that are ready to be executed;

a set of memory locations for storing a list of unavailable threads that are not ready to be executed; and

context event switching logic fed by signals from a plurality of shared resources with the signals causing the context event logic to indicate which threads are either available or unavailable for execution.

24 The processor of claim 23 wherein execution of a context swap instruction causes a currently running thread to be swapped out to the unavailable thread memory set and a thread from the available thread memory set to begin execution within a single execution cycle.

25. The processor of claim 23 wherein execution of a context swap instruction specifies one of the signal inputs and upon receipt of the specified signal input causes the swapped out thread to be stored in the available thread memory set.

26. The processor of claim 23 wherein execution of a context swap instruction specifies a defer_one operation which causes execution of one more instruction and then causes the current context to be swapped out.

27. The processor of claim 23 wherein the context event switching logic further comprises:

a ctx_enable register and with execution of a context swap instruction specifying a kill operation causing the ctx_enable bit to be set to indicate that this thread is not available for execution until the bit is cleared by another instruction.

28. A computer program product residing on a computer readable medium for causing a processor to perform a function comprises instructions causing the processor to:

perform a context swapping operation to cause a currently running thread to be swapped out to an unavailable thread memory set and a thread from an available thread memory set to begin execution within a single execution cycle.

29. The product of claim 28 wherein a context swapping operation specifies a signal input and upon receipt of the specified signal input causes the swapped out thread to be stored in the available thread memory set. .